Prof.
Rafael
Lima

UFCG
Universidade Federal
de Campina Grande

Introdução a VERILOG

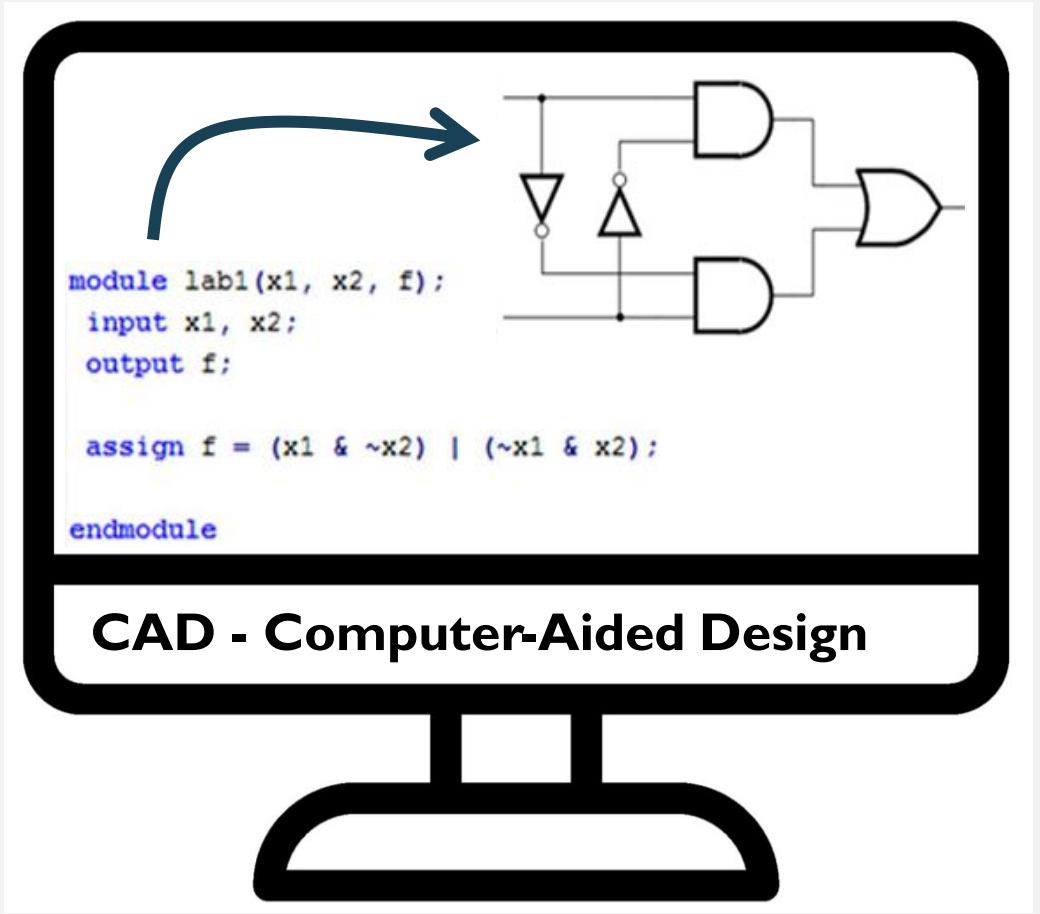
Visão geral da linguagem

Prof.
Rafael
Lima

O que é Verilog ?

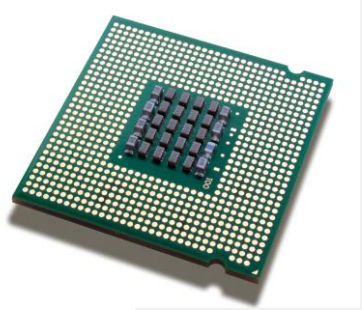
É uma Linguagem de descrição de Hardware (HDL)

IEEE Standard 1364-2005



Muito Importante!

NÃO confundir com linguagens de programação

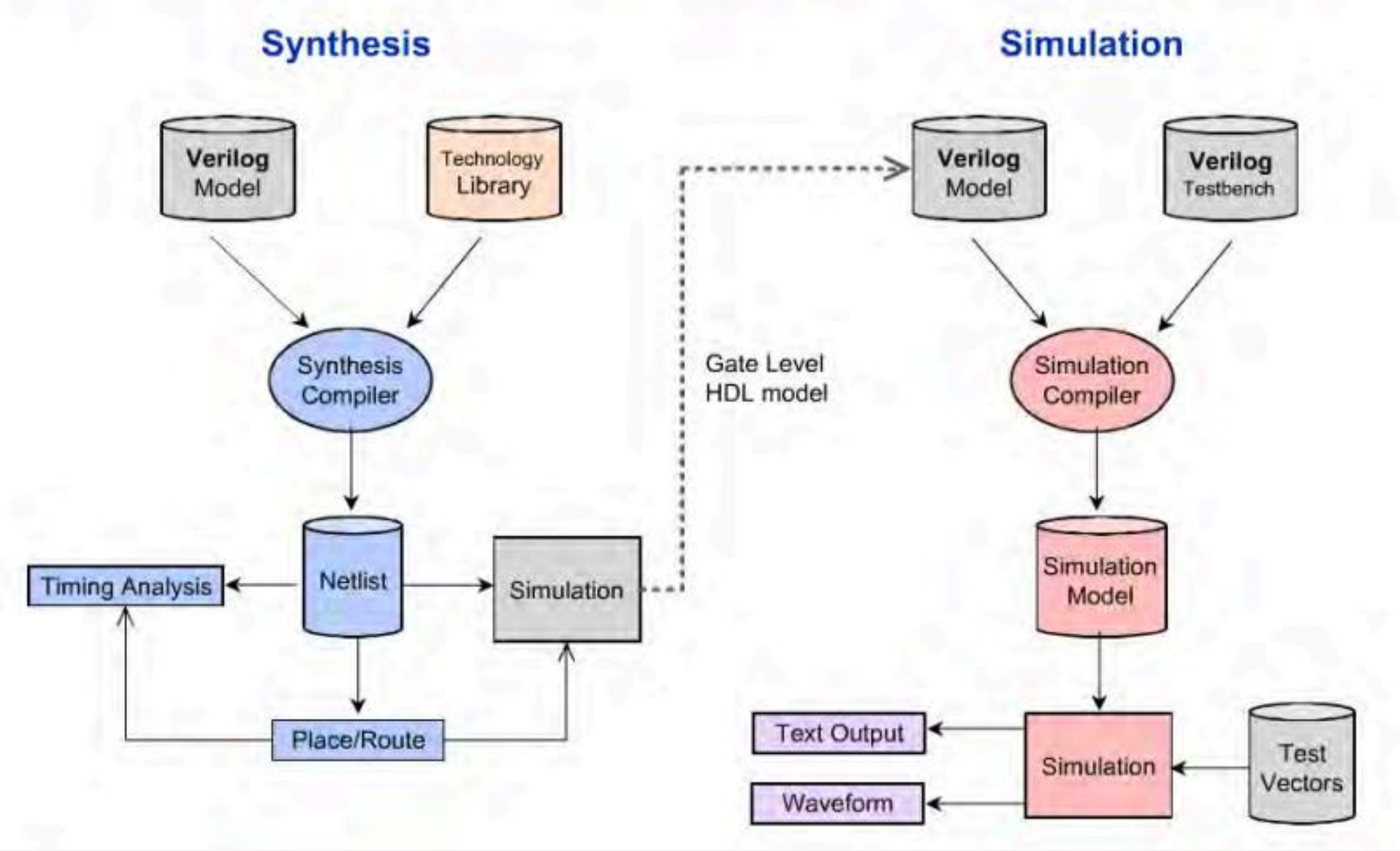


```
#include<stdio.h>
int main(void)
{
    printf("Hello World");
}
```

- Nível de abstração
- Produtividade

Prof. Rafael Lima

Síntese X Simulação

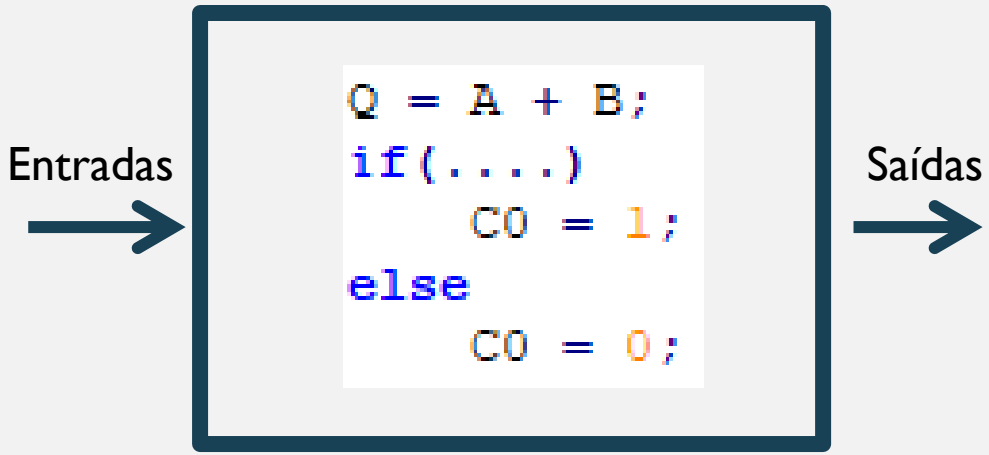


Prof. Rafael Lima

Behavior mod. X Structural mod.

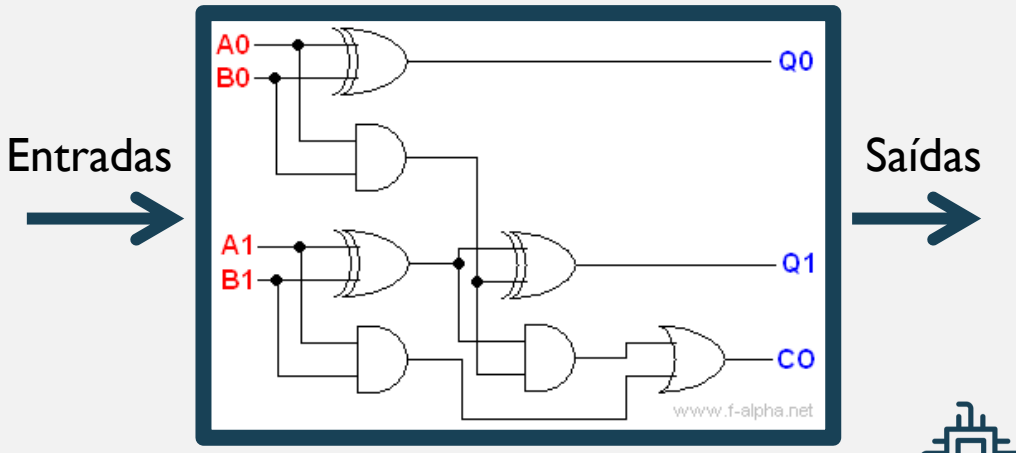
Behavior

- Descreve somente as funcionalidades entrada/saída
- A estrutura interna fica a cargo da ferramenta de síntese



Structural

- Define as funcionalidades e estruturas internas dos circuitos
- Estruturas de hardware são especificadas explicitamente



Prof. Rafael Lima

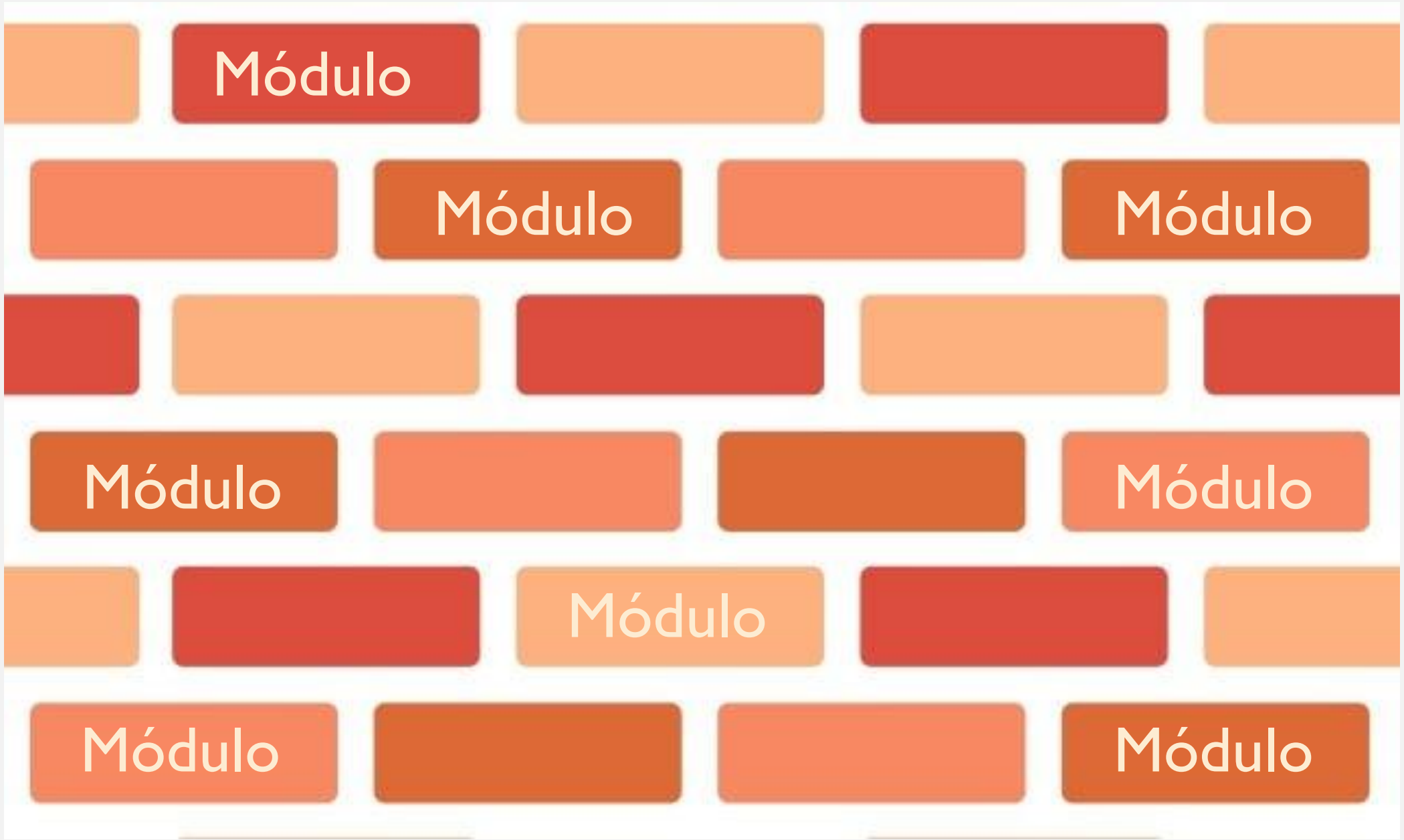
Prof.
Rafael
Lima

Introdução a VERILOG

Estrutura dos Módulos

Universidade Federal
de Campina Grande

Prof.
Rafael
Lima



Módulos

```
module nome_modulo (lista de portas);
...
//Declarações de variáveis/nets
/*Lógica do
circuito*/
//Case-sensitive e espaços são ignorados
...
endmodule
```

Tipos de portas

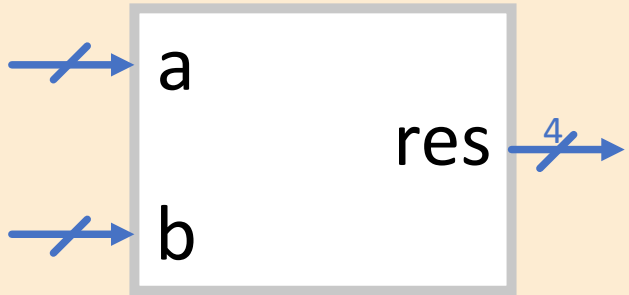
- **input**
- **output**
- **inout**

Declaração de portas

- <**Tipo_porta**> <Nome_porta>

Exemplo: Somador de 4bits

```
module somador
(input [3:0] a, b,
output [3:0] res);
...
//Implementação da lógica
...
endmodule
```



Prof. Rafael Lima

Data Types

Net

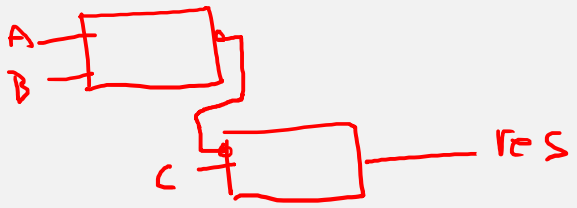
- Representa conexões físicas entre componentes

wire: representa um fio ou um nó

tri: nó tri-state

suply0: valor lógico 0

suply1: valor lógico 1



Variable

- Elemento para armazenamento temporário de dados

reg: variável de qualquer comprimento, sem sinal

integer: variável de 32 bits, sem sinal

Real, time, realtime: Não sintetizável

signed: modificador para indicar sinal!

Prof. Rafael Lima

Arrays

Declaração de Arrays

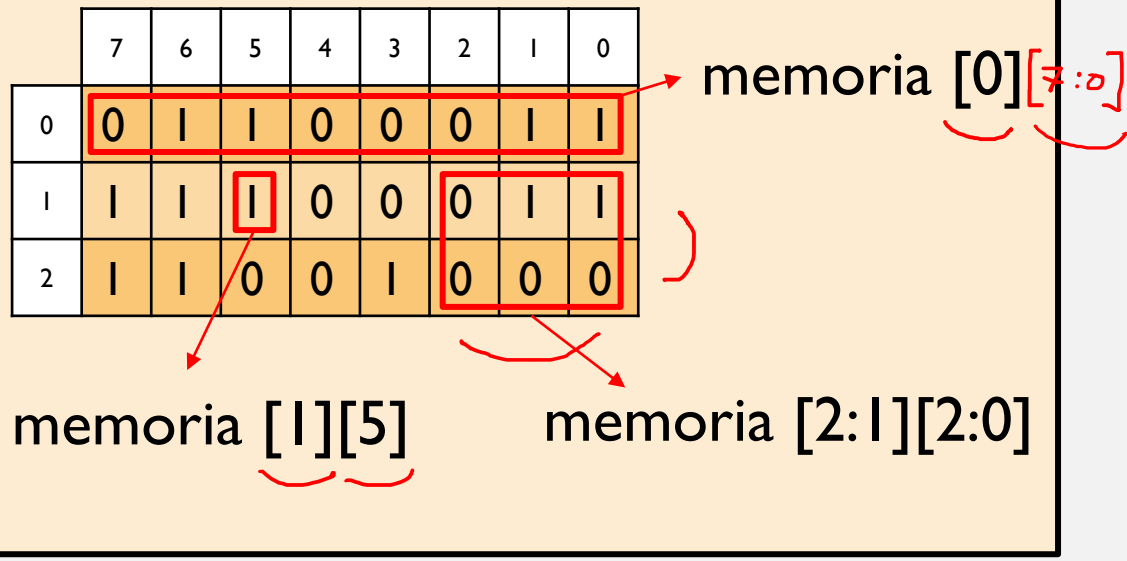
- <**DataType**> [a:b] nome [c:d]

Acesso à Arrays

- nome [c:d] [a:b]

Exemplo: Array de 3 bytes

Reg [7:0] memoria [2:0]



Prof. Rafael Lima

Atribuições

- **Atrib. Contínua** (**assign**)
- É sempre ativa

```
assign fio1 = fio2;
```

Circuitos combinacionais

Modelados em apenas 1 linha

```
assign fio1 = fio2 & fio3
```

- **Atrib. Procedural** (**always**)
- Ativa de acordo com a lista de sensibilidade
- **initial**: não sintetizável

```
always @ (fio2)
    reg1 = fio2;
```

Circuitos combinacionais

```
always @ (fio1, fio2...)
always @ (*)
```

Circuitos sequenciais

```
always @ (posedge clk, negedge reset)
begin
    ...
end
```

Prof. Rafael Lima

Atribuições

- **Atrib. Contínua (assign)**

Exemplo: Somador de 4bits

```
module somador
(input [3:0] a, b,
output [3:0] res);
    assign res = a+b;
endmodule
```

- **Atrib. Procedural (always)**

```
module somador
(input [3:0] a, b,
output reg [3:0] res);
    always @ (*)
        res = a+b;
endmodule
```

	LHS		RHS
assign	wire	=	wire/reg
always	reg	=	wire/reg

Begin ... End Equivalente as {} em C. Agrupamento de comandos em mais de uma linha

Prof. Rafael Lima

Atribuições

Blocking (=)

- Executada linha a linha, na sequência

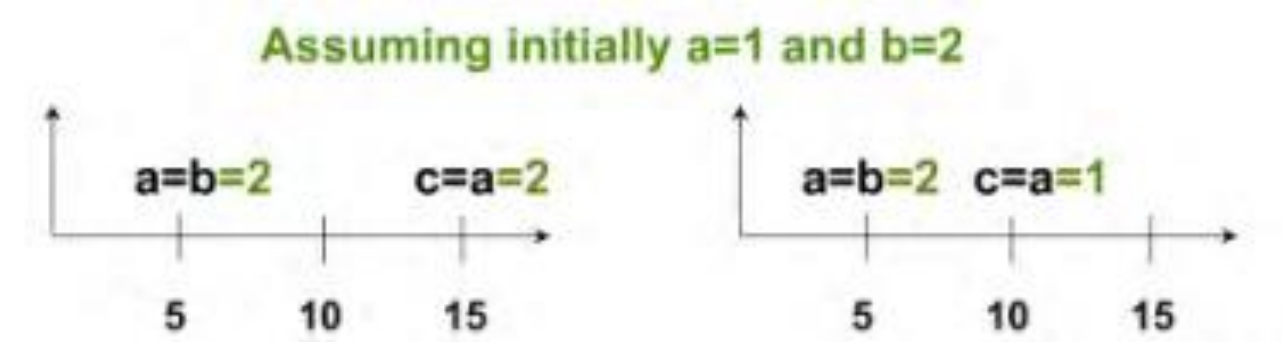
```
always @ (*)
begin
      a = #5 b;
      c = #10 a;
end
```

Nonblocking (<=)

- O lado direito de todas as atribuições são avaliados no mesmo instante

```
always @ (*)
begin
      a <= #5 b;
      c <= #10 a;
end
```

Recomendação:
= Circ. Combinatório
<= Circ. Sequencial



Prof. Rafael Lima

Constantes Numéricas

<tamanho(quant. de **BITs**)>'<base numérica> <valor da constante>

- Base default: **decimal**
- Tamanho default: **32 bits**

Decimal: **d**
Hexa: **h**
Binário: **b**
Octal: **o**
Signed: **s**

Exemplo:

3'b010	**decimal: 2**
8'd25	**decimal: 25**
16'hA_B_C_D	**decimal: 43981**
010	**decimal: 10**
3'd008	**decimal: ???**
-8'd3 (Complemento de 2 do núm. 3)	**decimal: ???**

comp 1 → 8'b0000_0011
+1 → 8'b1111_1100
decimal → 8'b1111_1101
8'd 253

Prof. Rafael Lima

Prof.
Rafael
Lima

Universidade Federal
de Campina Grande

Introdução a VERILOG

Instanciação de Módulos

Prof.
Rafael
Lima

Instanciação de Módulos

Módulo X

```
module ModuloX
(input a, b,
output c,...,d);
    ....
endmodule
```


```
ModuloX Instancia1 (.a(fio_a), .b(fio_b), .c(fio_c) .... );
ModuloX Instancia2 (.a(fio_a), .b(fio_b), .c(fio_c) .... );
                    ...
ModuloX InstanciaN (.a(fio_a), .b(fio_b), .c(fio_c) .... );
```

Prof. Rafael Lima

Instanciação de Módulos

```
module somador
(input [3:0] a, b,
output [3:0] res);
      assign res = a+b;
endmodule
```

- Sempre instanciar pelo NOME das portas

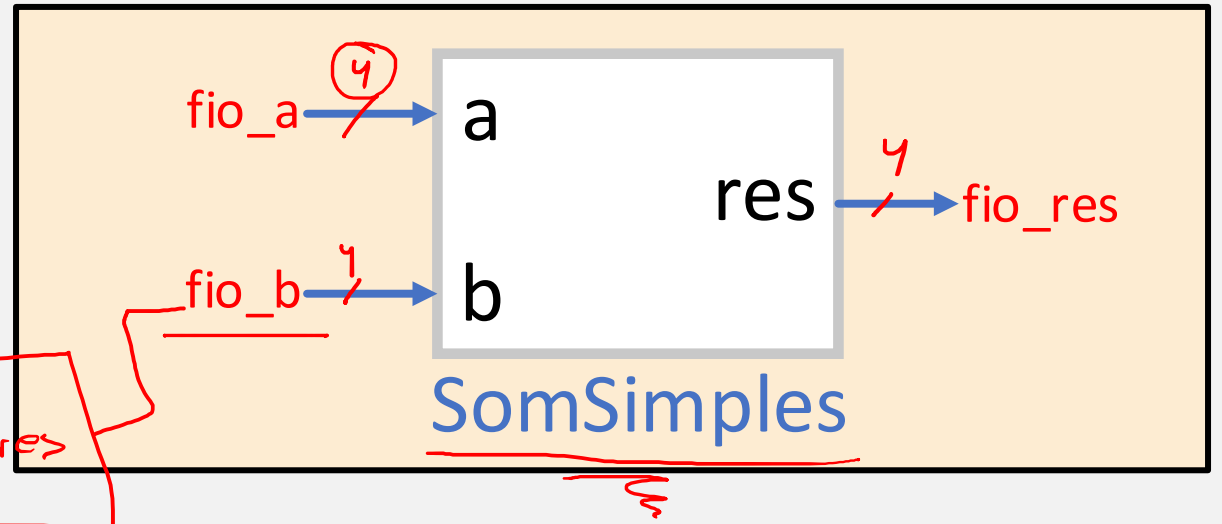
- <"tipo"_do_modulo>
  <nome_da_Instância>
  (.nome_da_porta1(fio1_conectado),
  .nome_da_porta2(fio2_conectado), ... ,
  .nome_da_portan(fion_conectado));

- A ordem das portas não importa!

Exemplo: Somador de 4bits

```
wire [3:0] fio_a, fio_b, fio_res;

Somador SomSimples (.a(fio_a), .b(fio_b), .res(fio_res));
```



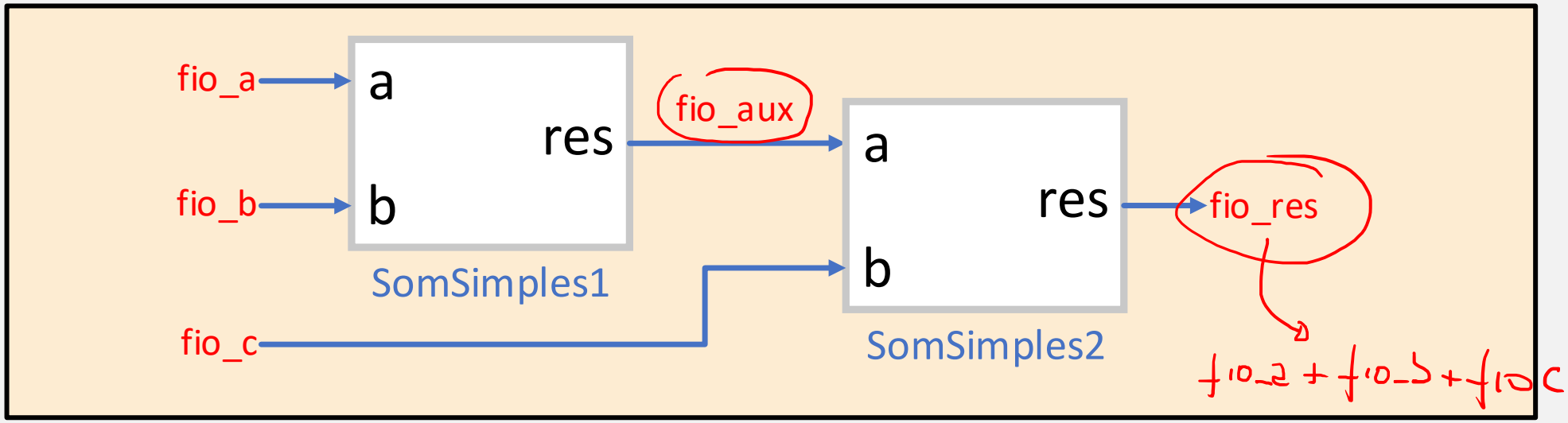
Prof. Rafael Lima

Instanciação de Módulos

Exemplo: Somador de 4bits com 3 entradas

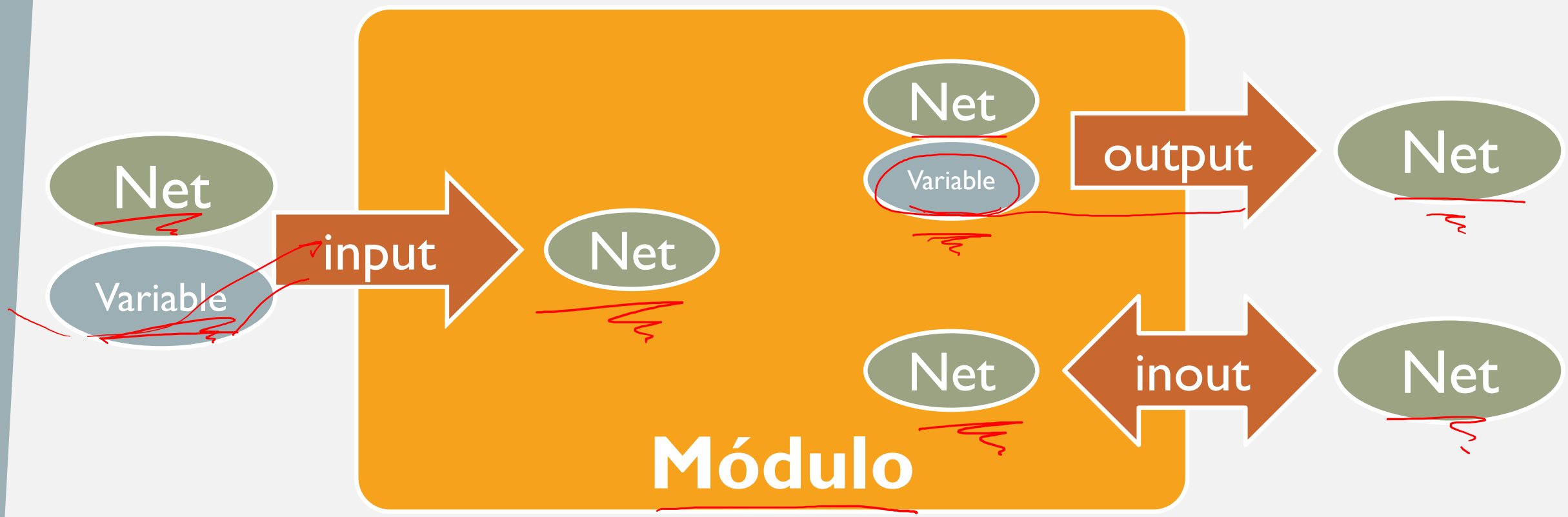
```
wire [3:0] fio_a, fio_b, fio_c, fio_res, fio_aux;

Somador SomSimples1 (.a(fio_a),.b(fio_b),.res(fio_aux));
Somador SomSimples2 (.a(fio_aux), .b(fio_c), .res(fio_res));
```



Prof. Rafael Lima

Instanciação de Módulos



Prof.
Rafael
Lima

Universidade Federal
de Campina Grande

Introdução a VERILOG

Operadores e Controle de fluxo

Prof.
Rafael
Lima

Operadores

Operator Symbol	Functionality	Examples ain = 5 ; bin = 10 ; cin = 2'b01 ; din = 2'b0z		
+	Add, Positive	bin + cin ⇒ 11	+bin ⇒ 10	ain + din ⇒ x
–	Subtract, Negate	bin – cin ⇒ 9	-bin ⇒ -10	ain – din ⇒ x
*	Multiply	ain * bin ⇒ 50		
/	Divide	bin / ain ⇒ 2		
%	Modulus	bin % ain ⇒ 0		
**	Exponent*	ain ** 2 ⇒ 25		

Operator Symbol	Functionality	Examples ain = 3'b101 ; bin = 3'b110 ; cin = 3'b01x	
~	Invert each bit	~ain ⇒ 3b'010	~cin ⇒ 3'b10x
&	**AND** each bit	ain & bin ⇒ 3'b100	bin & cin ⇒ 3'b010
|	**OR** each bit	ain | bin ⇒ 3'b111	bin | cin ⇒ 3'b11x
^	**XOR** each bit	ain ^ bin ⇒ 3'b011	bin ^ cin ⇒ 3'b10x
^~ or ~^	**XNOR** each bit	ain ^~ bin ⇒ 3'b100	bin ~^ cin ⇒ 3'b01x

Prof. Rafael Lima

Operadores

Operator Symbol	Functionality	Examples ain = 3'b101 ; bin = 3'b110 ; cin = 3'b01x	
>	Greater than	ain > bin ⇒ 1'b0	bin > cin ⇒ 1'bx
<	Less than	ain < bin ⇒ 1'b1	bin < cin ⇒ 1'bx
>=	Greater than or equal to	ain >= bin ⇒ 1'b0	bin >= cin ⇒ 1'bx
<=	Less than or equal to	ain <= bin ⇒ 1'b1	bin <= cin ⇒ 1'bx

Operator Symbol	Functionality	Examples ain = 3'b101 ; bin = 3'b110 ; cin = 3'b01x	
==	Equality	ain == bin ⇒ 1'b0	cin == cin ⇒ 1'bx
!=	Inequality	ain != bin ⇒ 1'b1	cin != cin ⇒ 1'bx
===	Case equality	ain === bin ⇒ 1'b0	cin === cin ⇒ 1'b1
!==	Case inequality	ain !== bin ⇒ 1'b1	cin !== cin ⇒ 1'b0

Prof. Rafael Lima

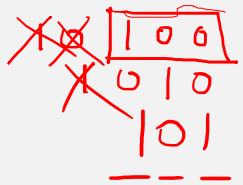
Operadores

Operator Symbol	Functionality	Examples ain = 3'b101 ; bin = 3'b000 ; cin = 3'b01x		
!	Expression not true	!ain ⇒ 1'b0	!bin ⇒ 1'b1	!cin ⇒ 1'bx
&&	AND of two expressions	ain && bin ⇒ 1'b0	bin && cin ⇒ 1'bx	
||	OR of two expressions	ain || bin ⇒ 1'b1	bin || cin ⇒ 1'bx	

Operator Symbol	Functionality	Examples ain = 4'b1010 ; bin = 4'b10xz ; cin = 4'b111z		
&	AND all bits	&ain ⇒ 1'b0	&bin ⇒ 1'b0	&cin ⇒ 1'bx
~&	NAND all bits	~&ain ⇒ 1'b1	~&bin ⇒ 1'b1	~&cin ⇒ 1'bx
|	OR all bits	|ain ⇒ 1'b1	|bin ⇒ 1'b1	|cin ⇒ 1'b1
~|	NOR all bits	~|ain ⇒ 1'b0	~|bin ⇒ 1'b0	~|cin ⇒ 1'b0
^	XOR all bits	^ain ⇒ 1'b0	^bin ⇒ 1'bx	^cin ⇒ 1'bx
^~ or ~^	XNOR all bits	~^ain ⇒ 1'b1	~^bin ⇒ 1'bx	~^cin ⇒ 1'bx

Prof. Rafael Lima

Operadores



Operator Symbol	Functionality	Examples ain = 3'b101 ; bin = 3'b01x	
<<	Logical shift left	ain << 2 ⇒ 3'b100	bin << 2 ⇒ 3'bx00
>>	Logical shift right	ain >> 2 ⇒ 3'b001	bin >> 2 ⇒ 3'b000
<<<	Arithmetic shift left	ain <<< 2 ⇒ 3'b100	bin <<< 2 ⇒ 3'bx00
>>>	Arithmetic shift right	ain >>> 2 ⇒ 3'b111 (signed)	bin >>> 2 ⇒3'b000 (signed)

Operator Symbol	Functionality	Format & Examples
?:	Conditional test	(condition) ? true_value : false_value sig_out = (sel == 2'b01) ? a : b
{ }	Concatenate	ain = 3'b010 ; bin = 3'110 {ain, bin} ⇒ 6'b010110
{ { } }	Replicate	{3 {3'b101} } ⇒ 9'b101101101

Prof. Rafael Lima

Controle de fluxo

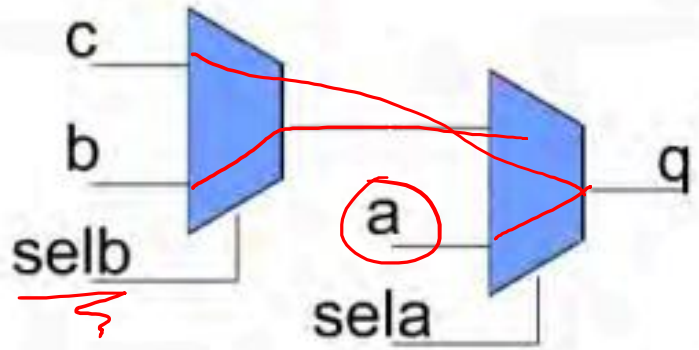
- **If - else**

Format:

```
if <condition1>
    {sequence of statement(s)}
else if <condition2>
    {sequence of statement(s)}
    ...
else
    {sequence of statement(s)}
```

Example:

```
always @ * begin
    if (sela)
        q = a;
    else if (selb)
        q = b;
    else
        q = c;
end
```



Sempre dentro de um Always

Controle de fluxo

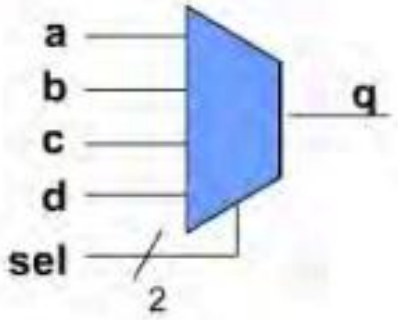
- **Case**

Format:

```
case {expression}
    <condition1> :
        {sequence of statements}
    <condition2> :
        {sequence of statements}
    ...
    default : -- (optional)
        {sequence of statements}
endcase
```

Example:

```
always @ * begin
    case (sel)
        2'b00 : q = a;
        2'b01 : q = b;
        2'b10 : q = c;
        default : q = d;
    endcase
end
```



Sempre dentro de um Always

Controle de fluxo

- **Repeat**

■ **repeat** loop - executes a fixed number of times

```
if (rotate == 1)
    repeat (8) begin
        tmp = data[15];
        data = {data << 1, tmp};
    end
```

Repeats a rotate operation 8 times

Sempre dentro de um Always

Parâmetros

```
parameter size = 8;
reg [size-1:0] dataX;
```

Prof.
Rafael
Lima

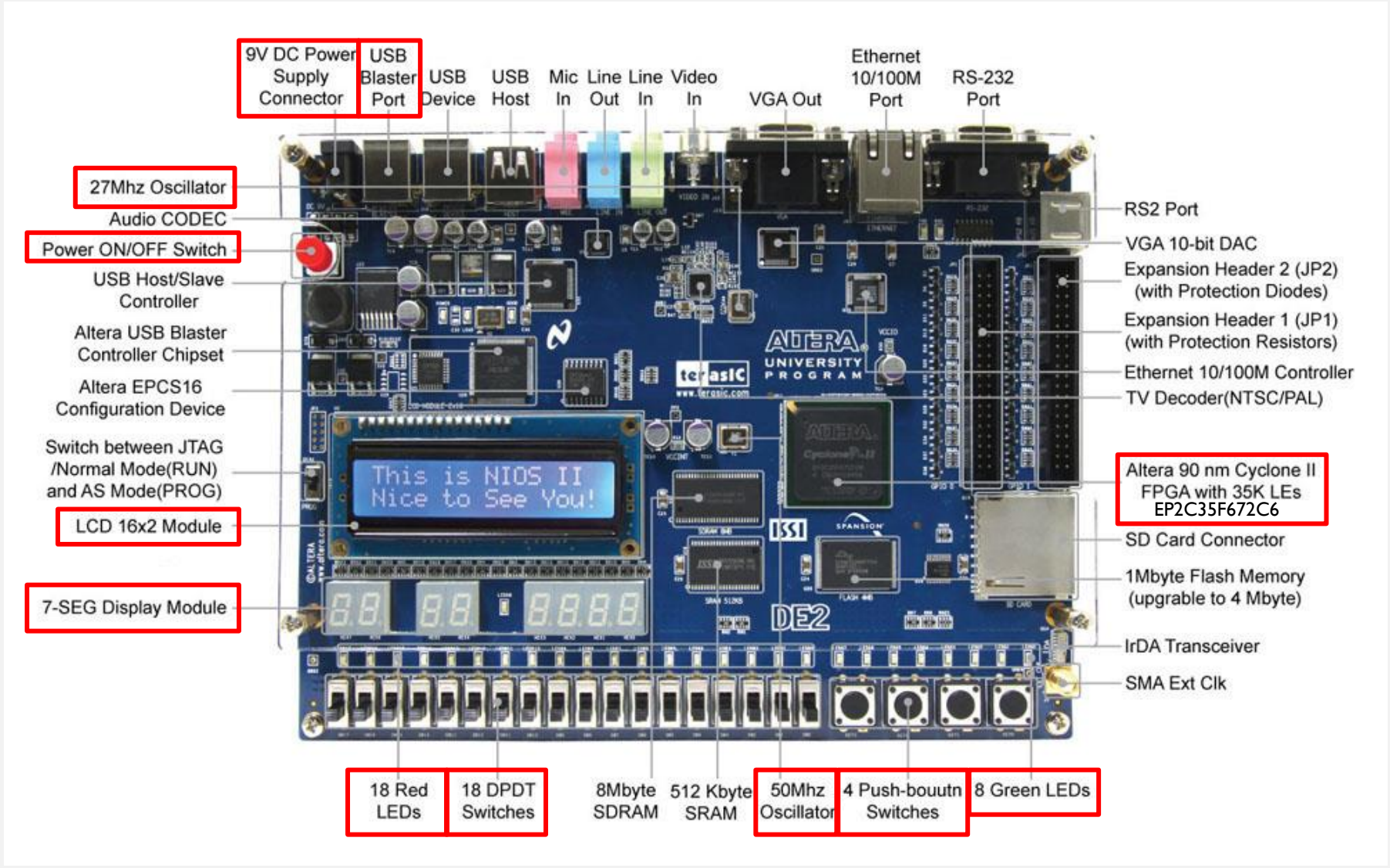
UFCG
Universidade Federal
de Campina Grande

Altera DE2

Placa FPGA

Prof.
Rafael
Lima

Placa Altera DE2

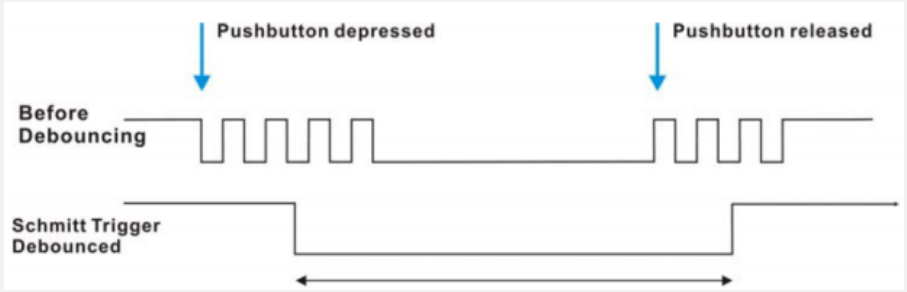


Prof. Rafael Lima

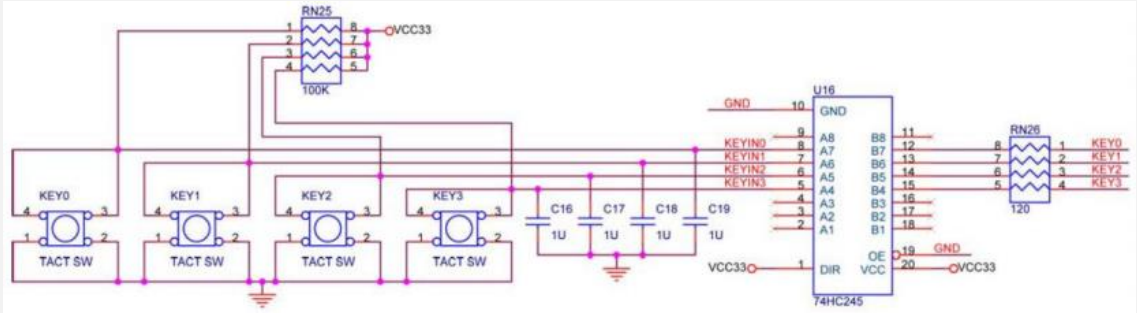
Placa Altera DE2

BOTÕES

- Total de 4 botões tácteis (com circuito de *debounce*).



- Identificação: KEY0 até KEY3.
- Nível BAIXO ativo (ao apertar o sinal é ZERO).

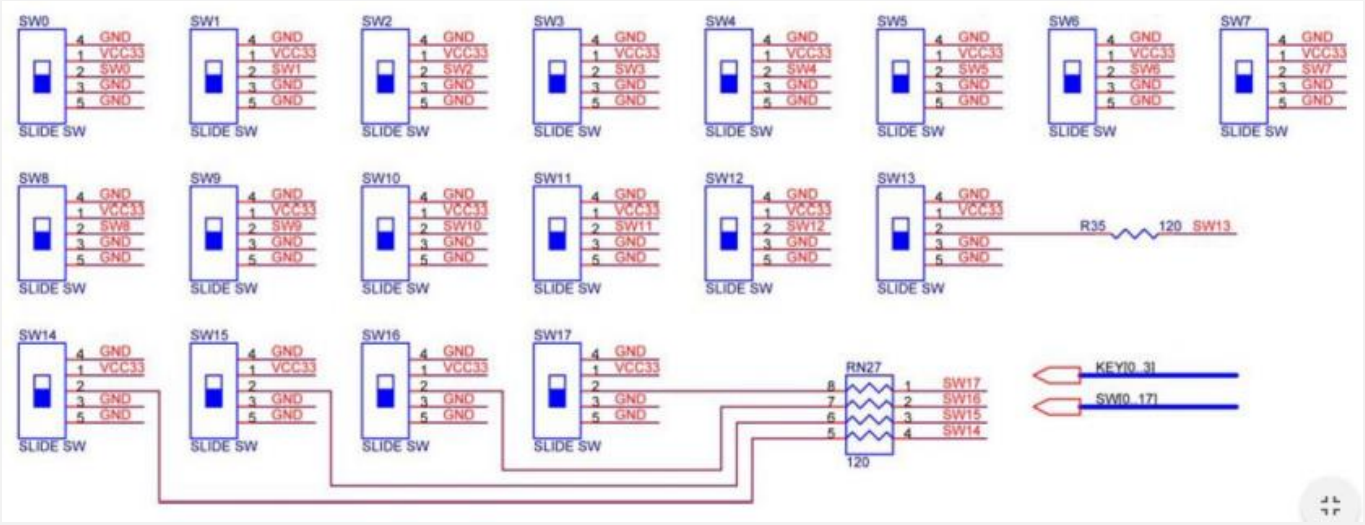


Prof. Rafael Lima

Placa Altera DE2

CHAVES

- 18 chaves (sem debounce)
  - Posição para baixo, sinal ZERO
  - Posição em alto, sinal UM
- Identificação: SW0 até SW17

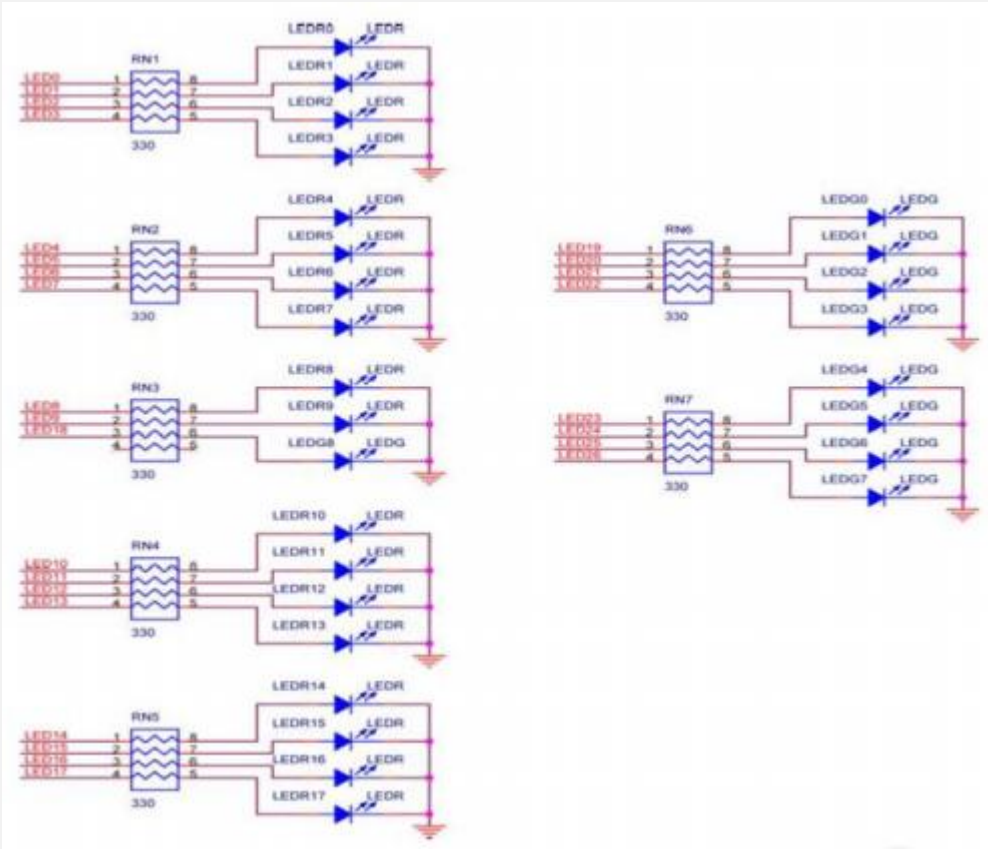


Prof. Rafael Lima

Placa Altera DE2

LEDs

- 18 leds vermelhos (LEDR0 até LEDR17)
- 9 leds verdes (LEDG0 até LEDG8)
- Sinais de leds ativos em ALTO.

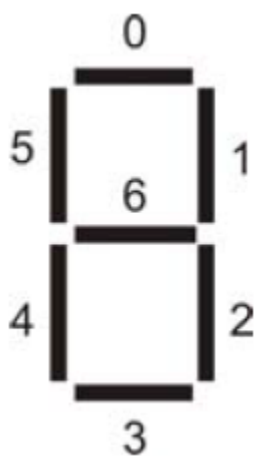


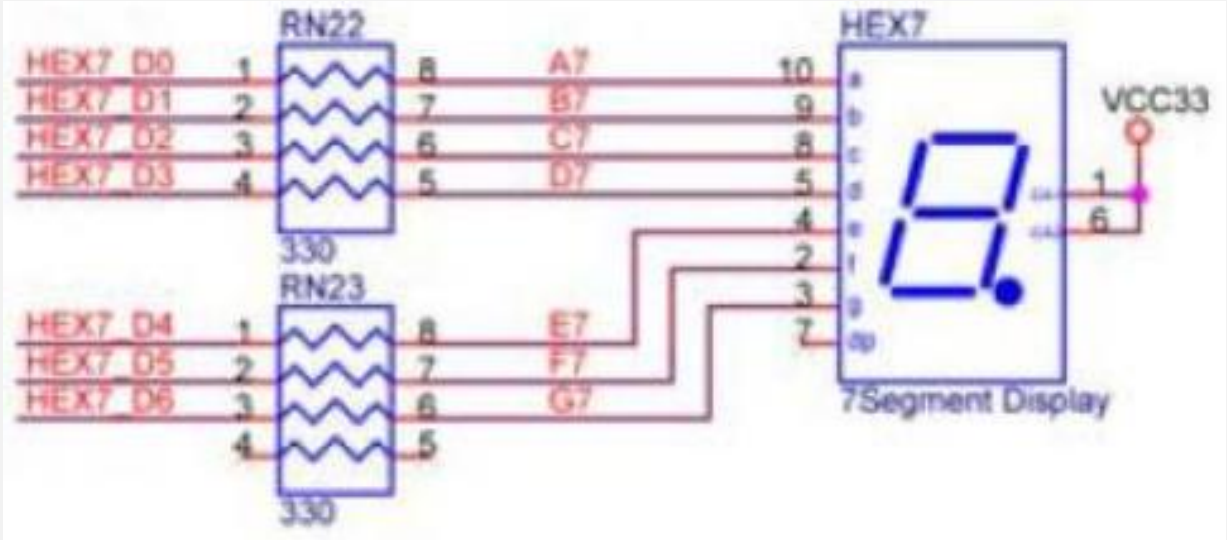
Prof. Rafael Lima

Placa Altera DE2

DISPLAYs de 7 SEGMENTOS

- 8 displays de 7 segmentos. – HEX0 até HEX7.
- Nível baixo ativo

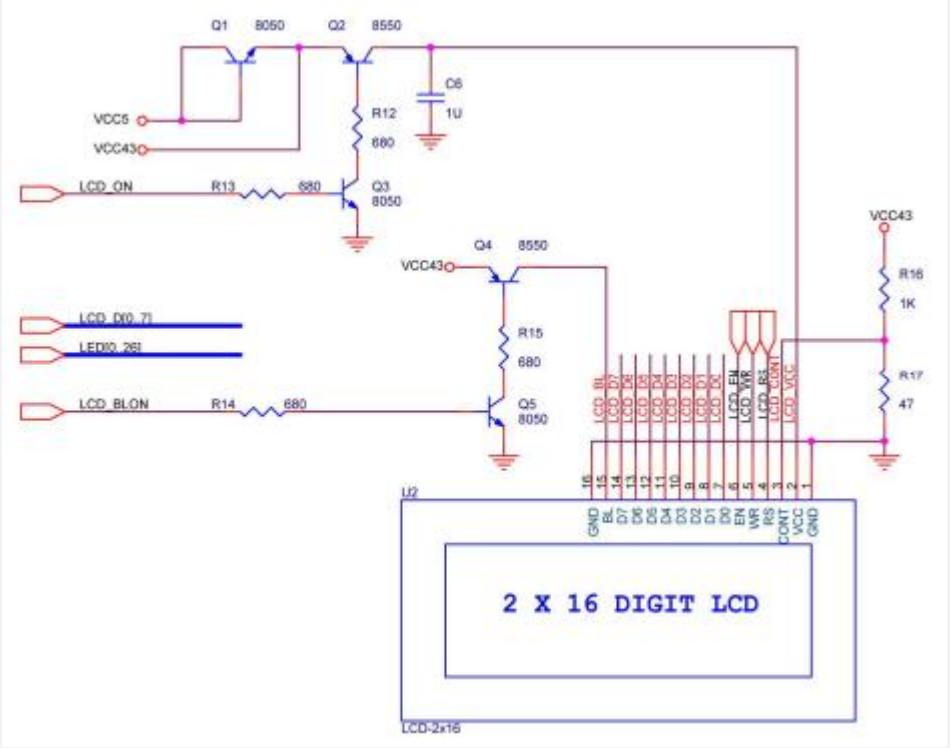




Placa Altera DE2

LCD

- LCD de 32 caracteres (2 linhas e 16 colunas)
- Biblioteca LCD_Controller.v e LCD_TEST2.v
- Caracteres agrupados de 2 em 2

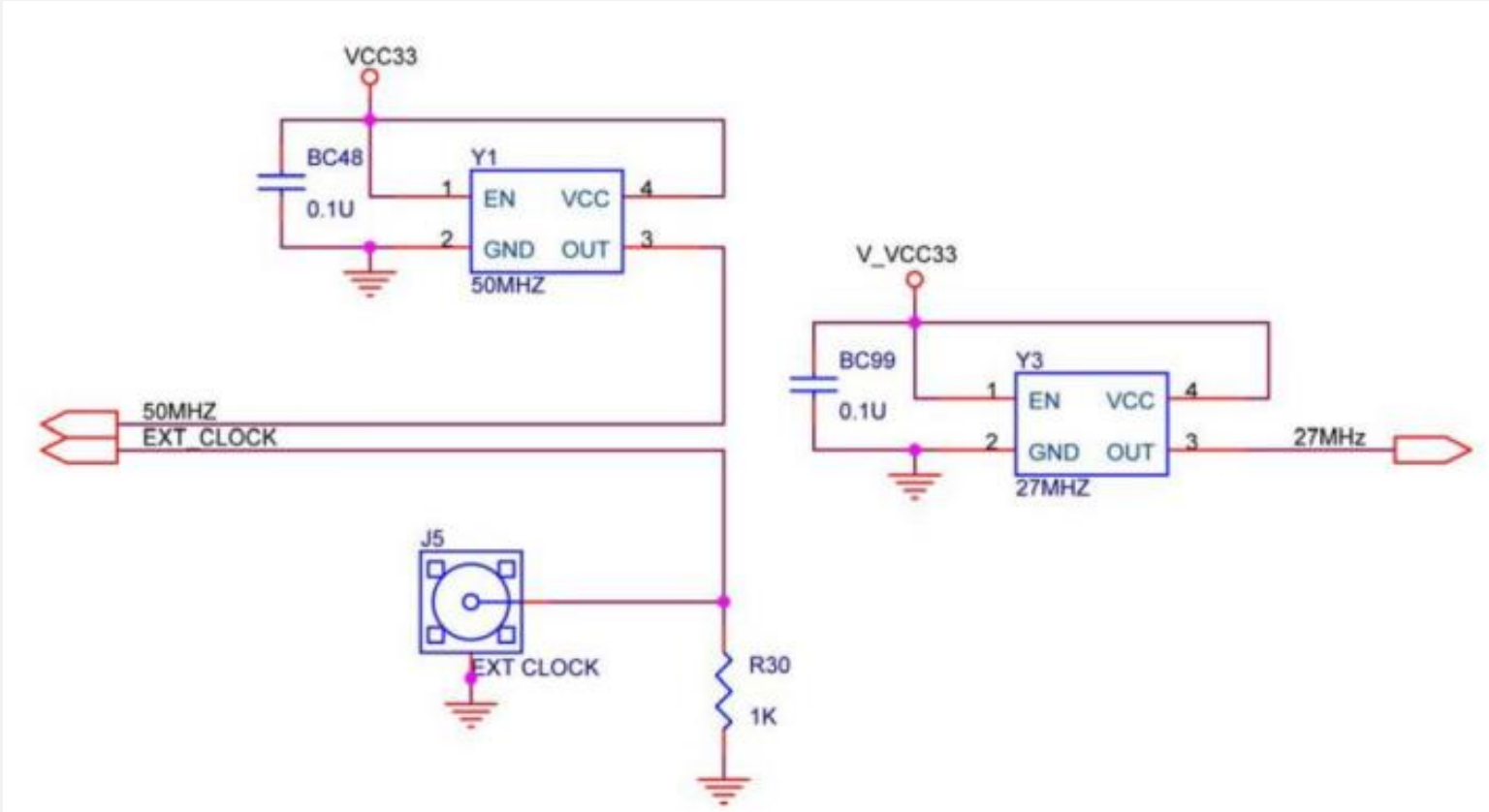


Prof. Rafael Lima

Placa Altera DE2

CLOCKs

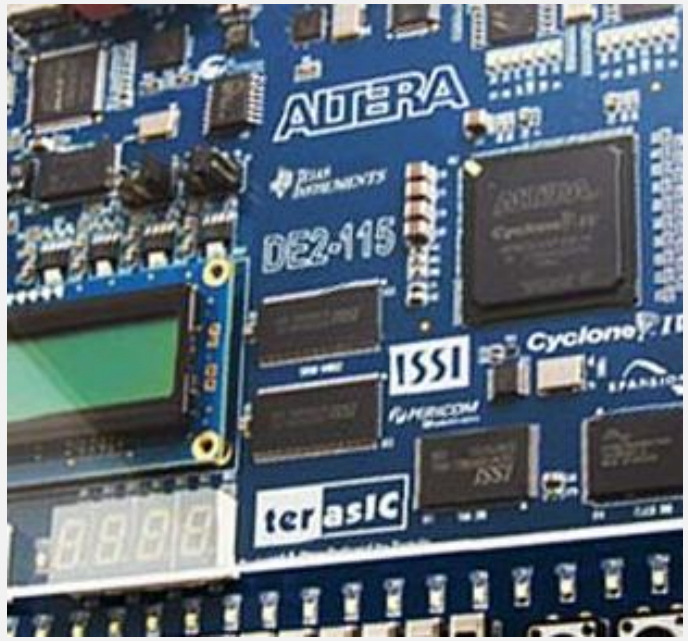
- Clocks internos:
  - 50 MHz (CLOCK_50)
  - 27 MHz (CLOCK_27)



Prof. Rafael Lima

Prof.
Rafael
Lima

UFCG
Universidade Federal
de Campina Grande



ALTERA®
QUARTUS® II

Introdução a VERILOG

Primeiro projeto Quartus II

Prof.
Rafael
Lima

Placa Altera DE2

- **Setup Experimental**